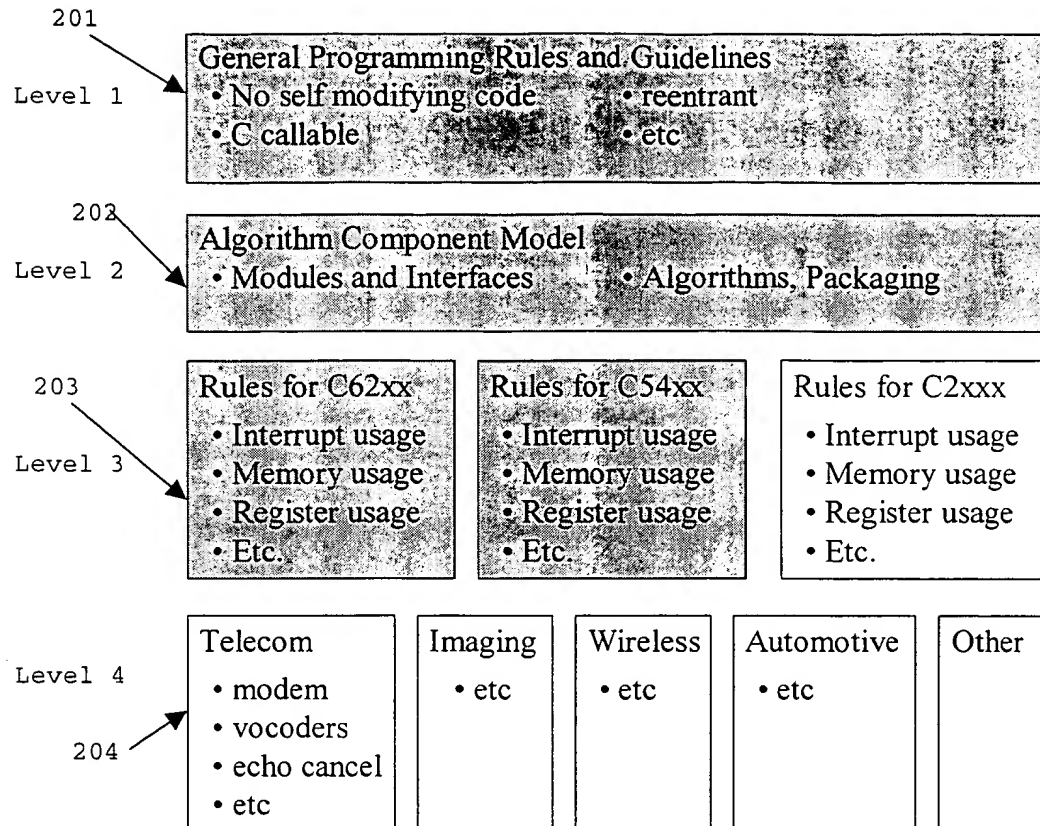


Figure 1A

000200 02999900





4/30

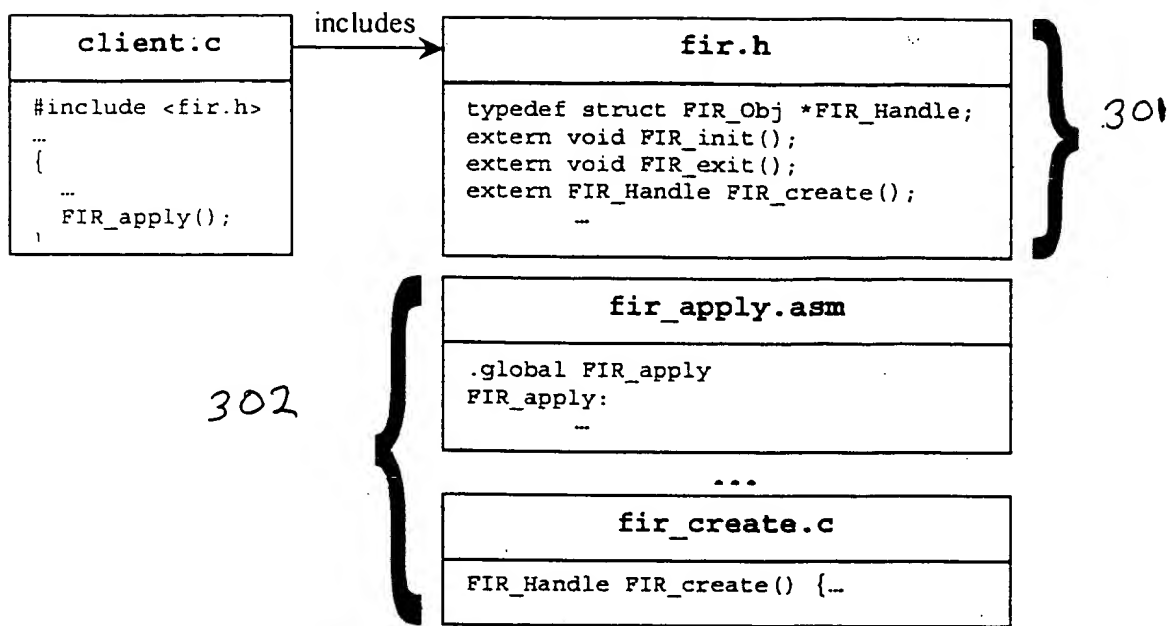


Figure 3

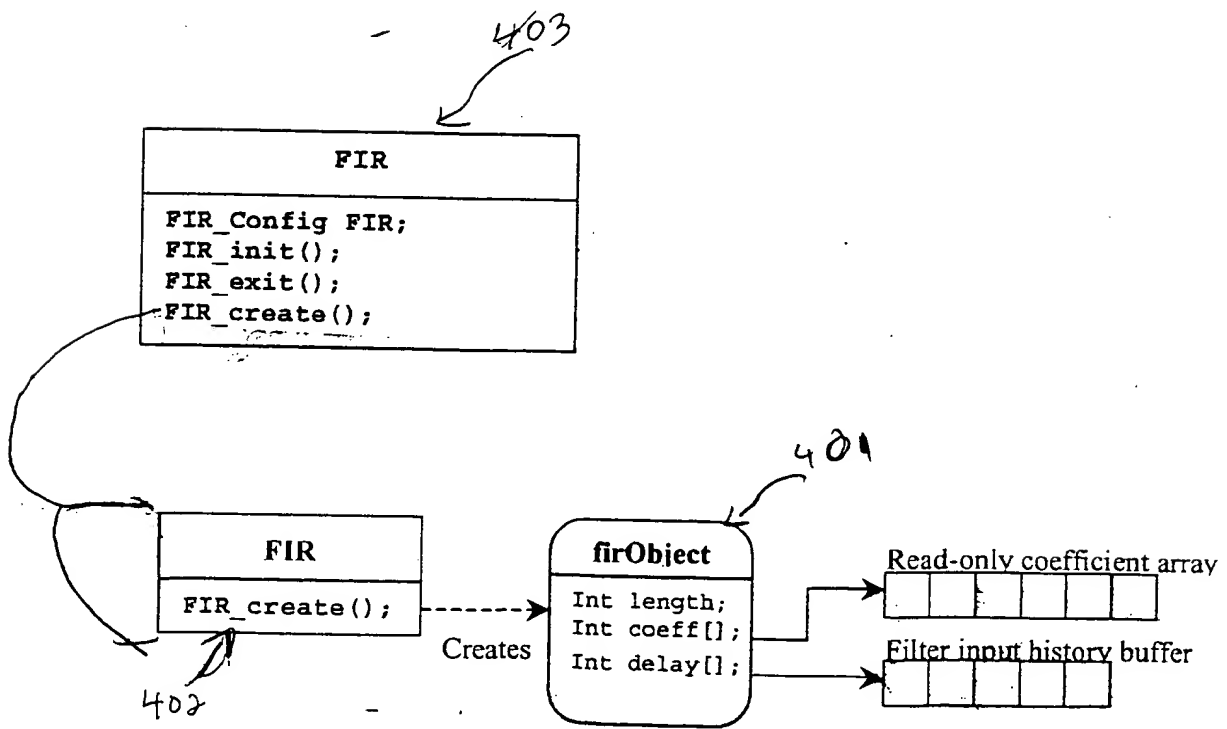


Figure 4

```
void FIR_init(void)
{
}

void FIR_exit(void)
{
}
```

501

```

typedef FIR_Params {          /* FIR_Obj creation parameters */
    int frameLen;             /* input/output frame length */
    int *coeff;               /* pointer to filter coefficients */
} FIR_Params;

505 FIR_Params FIR_PARAMS = { 64, NULL }; /* default parameters */

typedef struct FIR_Obj {      /* FIR_Obj definition */
    int hist[16];             /* previous input value */
    int frameLen;             /* input frame length */
    int *coeff;
} FIR_Obj;

```

```

502 {
    FIR_Handle FIR_create(FIR_Obj *fir, const FIR_Params *params)
    {
        if (fir != NULL) {
            if (params == NULL) { /* use defaults if params is NULL */
                params = &FIR_PARAMS;
            }
            fir->frameLen = params->frameLen;
            fir->coeff = params->coeff;
            memset(fir->hist, 0, sizeof (fir->hist));
        }

        return (fir);
    }
}

```

```
503 { void FIR_delete(FIR_Handle fir)
```

```

504 {
    void FIR_apply(FIR_Handle fir, int in[], int out[])
    {
        int i;

        /* filter data using coefficients fir->coeff and
           history fir->hist */
        for (i = 0; i < fir->frameLen; i++) {
            out[i] = filter(in[i], fir->coeff, fir->hist);
        }
    }
}

```



```

/*-----*/
/* TYPES AND CONSTANTS */
/*-----*/
#define IALG_DEFMEMRECS 4
#define IALG_OBJMEMREC 0
#define IALG_SYSCMD 256
#define IALG_EOK 0
#define IALG_EFAIL -1

/* default number of memory records */
/* memory record index of instance object */
/* minimum "system" IALG_Cmd value */
/* successful return status code */
/* unspecified error return status code */

/* scratch memory */
/* persistent memory */
/* write-once persistent memory */

/* program memory space bit */
/* external memory space bit */

/* external program memory */
/* internal program memory */
/* off-chip data memory (accessed sequentially) */
/* off-chip data memory (accessed randomly) */
/* dual access on-chip data memory */
/* dual access on-chip data memory */
/* single access on-chip data memory */
/* block 0, equivalent to IALG_SARAM */
/* block 1, if independant blocks required */

/* 'sizeof' memory request in MAUs (minimum address-able unit) */
/* alignment requirement (in MAUs) */
/* allocation space */
/* memory attributes */
/* base address of allocated buf */

```

Figure 7A

7/30

```

/*
 * ===== IALG_Obj =====
 * Algorithm instance object definition
 *
 * All XDAIS algorithm instance objects *must* have this structure as their first element. However, they do not
 * need to initialize it; initialization of this sub-structure is done by the "framework".
 */
105 { typedef struct IALG_Obj {
        struct IALG_Fxns *fxns;
    } IALG_Obj;

/*
 * ===== IALG_Handle =====
 * Handle to an algorithm instance object
 */
107 { typedef struct IALG_Obj *IALG_Handle;

/*
 * ===== IALG_Params =====
 * Algorithm instance creation parameters
 * All XDAIS algorithm parameter structures *must* have a this as their first element.
 */
103 { typedef struct IALG_Params {
        Int size; /* number of MAUs (i.e. the 'sizeof') the structure */
    } IALG_Params;

/*
 * ===== IALG_Status =====
 * Pointer to algorithm specific status structure
 * All XDAIS algorithm status structures *must* have this as their first element.
 */
104 { typedef struct IALG_Status {
        Int size; /* number of MAUs (i.e. the 'sizeof') the structure */
    } IALG_Status;

/*
 * ===== IALG_Cmd =====
 * Algorithm specific command. This command is used in conjunction with IALG_Status to get and set algorithm
 * specific attributes via the algControl method.
 */
    typedef unsigned int IALG_Cmd;

```

Figure 7B

8/30



9/30

```
typedef struct IALG_Fxns {
    *ImplementationId;
    (*algActivate)(IALG_Handle);
    (*algAlloc)(const IALG_Params *, struct IALG_Fxns **, IALG_MemRec *);
    (*algControl)(IALG_Handle, IALG_Cmd, IALG_Status *);
    (*algDeactivate)(IALG_Handle);
    (*algFree)(IALG_Handle, IALG_MemRec *);
    (*algInit)(IALG_Handle, const IALG_MemRec *, IALG_Handle, const
    *);
    (*algMoved)(IALG_Handle, const IALG_MemRec *, IALG_Handle, const
    *);
    (*algNumAlloc)(Void);
} IALG_Fxns;
```

Figure 7C

10/30

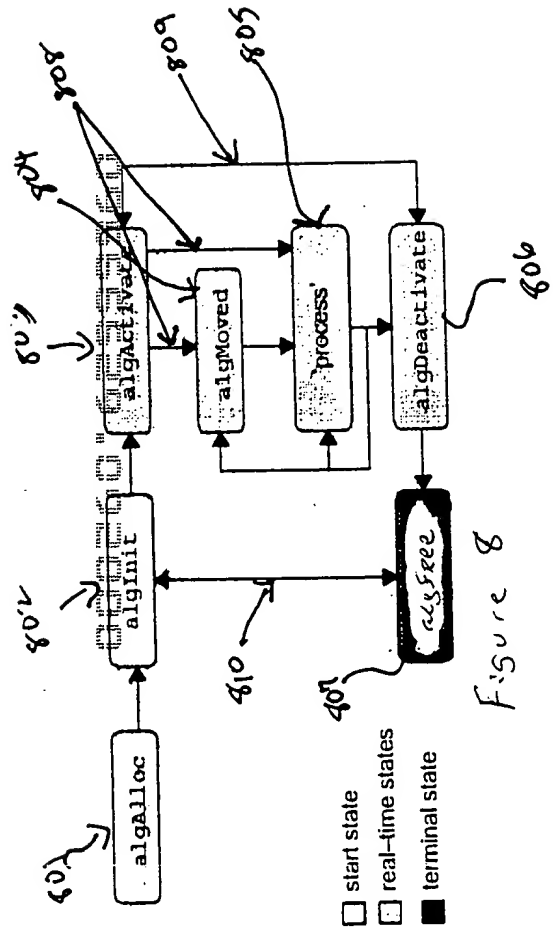


Figure 8

```

client()
{
    FIR_Params stdParams;
    FIR_TI_Params tiParams;

    stdParams = FIR_PARAMS;
    stdParams.coeff = ...;
    fxns->algAlloc(&stdParams, ...);

    tiParams = FIR_TI_PARAMS;
    tiParams.coeff = ...;
    fxns->algAlloc(&tiParams, ...);

    Int FIR_TI_algAlloc(IALG_Params *clientParams, ...)
    {
        FIR_TI_Params params = FIR_TI_PARAMS;

        /* client passes in parameters, use them to override defaults */
        if (clientParams != NULL) {
            memcpy(&params, clientParams, clientParams->size);
        }

        /* use params as the complete set of parameters */
    }
}

```

Figure 9

11/30

```
#define MAXMEMRECS 16
```

```
typedef struct ALG_Obj {
    IALG_Fxns    fxns;      /* algorithm functions */
} ALG_Obj;
```

```
IALG_Handle ALG_create(IALG_Fxns *fxns, IALG_Params *params)
```

```
{
    IALG_MemRec  memTab[MAXMEMRECS];
    IALG_Handle  alg = NULL;
    Int          n;

    if (fxns->algNumAlloc() <= MAXMEMRECS) {
        n = fxns->algAlloc(params, memTab);
        if (allocMemory(memTab, n)) {
            alg = (IALG_Handle)memTab[0].base;
            alg->fxns = fxns;
            if (fxns->algInit(alg, memTab, params) != IALG_EOK) {
                fxns->algFree(alg, memTab);
                freeMemory(memTab, n);
                alg = NULL;
            }
        }
    }
}
```

```
return (alg);
```

```
Void ALG_delete(IALG_Handle alg)
```

```
{
    IALG_MemRec memTab[MAXMEMRECS];
    Int n;

    n = alg->fxns->algFree(alg, memTab);
    freeMemory(memTab, n);
}
```

Figure 10

```

Void FIR_apply(FIR_Handle alg, Int *in[], Int *out[])
{
    /* do app specific initialization of scratch memory */
    if (alg->fxns->ialg.algActivate != NULL) {
        alg->fxns->ialg.algActivate(alg);
    }

    /* filter data */
    alg->fxns->filter(alg, in, out);

    /* do app specific store of persistent data */
    if (alg->fxns->ialg.algDeactivate != NULL) {
        alg->fxns->ialg.algDeactivate(alg);
    }
}

```

13110	13111	13112	13113	13114	13115	13116	13117	13118	13119	13120	13121	13122	13123	13124	13125	13126	13127	13128	13129	13130	13131	13132	13133	13134	13135	13136	13137	13138	13139	13140	13141	13142	13143	13144	13145	13146	13147	13148	13149	13150	13151	13152	13153	13154	13155	13156	13157	13158	13159	13160	13161	13162	13163	13164	13165	13166	13167	13168	13169	13170	13171	13172	13173	13174	13175	13176	13177	13178	13179	13180	13181	13182	13183	13184	13185	13186	13187	13188	13189	13190	13191	13192	13193	13194	13195	13196	13197	13198	13199	13200	13201	13202	13203	13204	13205	13206	13207	13208	13209	13210	13211	13212	13213	13214	13215	13216	13217	13218	13219	13220	13221	13222	13223	13224	13225	13226	13227	13228	13229	13230	13231	13232	13233	13234	13235	13236	13237	13238	13239	13240	13241	13242	13243	13244	13245	13246	13247	13248	13249	13250	13251	13252	13253	13254	13255	13256	13257	13258	13259	13260	13261	13262	13263	13264	13265	13266	13267	13268	13269	13270	13271	13272	13273	13274	13275	13276	13277	13278	13279	13280	13281	13282	13283	13284	13285	13286	13287	13288	13289	13290	13291	13292	13293	13294	13295	13296	13297	13298	13299	13300	13301	13302	13303	13304	13305	13306	13307	13308	13309	13310	13311	13312	13313	13314	13315	13316	13317	13318	13319	13320	13321	13322	13323	13324	13325	13326	13327	13328	13329	13330	13331	13332	13333	13334	13335	13336	13337	13338	13339	13340	13341	13342	13343	13344	13345	13346	13347	13348	13349	13350	13351	13352	13353	13354	13355	13356	13357	13358	13359	13360	13361	13362	13363	13364	13365	13366	13367	13368	13369	13370	13371	13372	13373	13374	13375	13376	13377	13378	13379	13380	13381	13382	13383	13384	13385	13386	13387	13388	13389	13390	13391	13392	13393	13394	13395	13396	13397	13398	13399	13400	13401	13402	13403	13404	13405	13406	13407	13408	13409	13410	13411	13412	13413	13414	13415	13416	13417	13418	13419	13420	13421	13422	13423	13424	13425	13426	13427	13428	13429	13430	13431	13432	13433	13434	13435	13436	13437	13438	13439	13440	13441	13442	13443	13444	13445	13446	13447	13448	13449	13450	13451	13452	13453	13454	13455	13456	13457	13458	13459	13460	13461	13462	13463	13464	13465	13466	13467	13468	13469	13470	13471	13472	13473	13474	13475	13476	13477	13478	13479	13480	1348
-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	------

```
typedef struct EncoderObj {
    IALG_Obj ialgObj;    /* IALG object MUST be first field */
    Int *workBuf;        /* pointer to on-chip scratch memory */
    Int *historyBuf;     /* previous frame's data in ext mem */
    ... ;
} EncoderObj;
```

```
EncoderObj *inst = (EncoderObj *)handle;
```

$$\left. \begin{array}{l} 1 \\ \end{array} \right\} 120$$

```
{
    EncoderObj *inst = (EncoderObj *)handle;
```

```
/* encode data */
```

}

```
EncoderObj *inst = (EncoderObj *)handle;
```

1202

[illegible]

```

typedef struct EncoderObj
    IALG_Obj   ialgObj
    Int        *workBuf;
    Int        workBufLen;
    ...;
) EncoderObj;

typedef struct EncoderParams {
    Int frameDuration;
};
EncoderParams ENCODERATTRS = {5};

Int algAlloc(IALG_Params *algParams, IALG_Fxns **p, IALG_MemRec memTab[])
{
    EncoderParams *params = (EncoderParams *)algParams;
    if (params == NULL) {
        params = &ENCODERATTRS;
    }
    memTab[0].size = sizeof (EncoderObj);
    memTab[0].alignment = 1;
    memTab[0].type = IALG_PERSIST;
    memTab[0].space = IALG_EXTERNAL;
    memTab[1].size = params->frameDuration * 8 * sizeof(int);
    memTab[1].alignment = 1;
    memTab[1].type = IALG_PERSIST;
    memTab[1].space = IALG_DARAM;
    return (2);
}

```

Fig. 13

14/30

15/30

### Example

```
typedef struct EncoderStatus {
    Bool voicePresent; /* voice in current frame? */
    ... ;
} EncoderStatus;

typedef enum {EncoderGetStatus, ...} EncoderCmd;

Void algControl(IALG_Handle handle,
                IALG_Cmd cmd, IALG_Status *status)
{
    EncoderStatus *sptr = (EncoderStatus *)status;

    switch ((EncoderCmd)cmd) {
        case EncoderGetStatus:
            sptr->voicePresent = ...;
            ...
        case EncoderSetMIPS:
            ...
    }
}
```

5.15. 14

```

typedef struct EncoderObj {
    IALG_Obj   ialgObj;
    Int        *workBuf;
    Int        workBufLen;
    ...;
} EncoderObj;
Int algFree(IALG_Handle handle, IALG_MemRec memTab[])
{
    EncoderObj *inst = (EncoderObj *)handle;

    algAlloc(NULL, memTab);

    memTab[1].size = inst->workBufLen * sizeof(Int);
    memTab[1].base = (Void *)inst->workBuf;
    return(2);
}

Int algAlloc(IALG_Params *params, IALG_MemRec memTab[])
{
    memTab[0].size = sizeof (EncoderObj);
    memTab[0].alignment = 1;
    memTab[0].type = IALG_PERSIST;
    memTab[0].space = IALG_EXTERNAL;

    memTab[1].size = 80;
    memTab[1].alignment = 1;
    memTab[1].type = IALG_PERSIST;
    memTab[1].space = IALG_DARAM;

    return (2);
}

```

Fig 15



```

00000000 00000000
typedef struct EncoderObj {
    IALG_Obj   ialgObj;
    Int        workBuf;
    Int        workBufLen;
    ...;
} EncoderObj;

Int algInit(IALG_Handle handle,
            IALG_MemRec memTab[], IALG_Handle p, IALG_Params *algParams)
{
    EncoderObj *inst = (EncoderObj *)handle;
    EncoderParams *params = (EncoderParams *)algParams;

    if (params == NULL) {
        params = &ENCODERATTRS;
    }

    inst->workBuf = memTab[1].base;
    inst->workBufLen = params->frameDuration * 8;
    ...
    return (IALG_EOK);
}

```

Fig 16

```
typedef struct EncoderObj {
    IALG_Obj ialgObj /* IALG object MUST be first field */
    int workBuf; /* pointer to on-chip scratch memory */
    Int workBufLen; /* workBuf length (in words) */
    ... ;
} EncoderObj;
```

```
algMoved(IALG_Handle handle,
         IALG_Params *algParams, IALG_MemRec memTab[])
```

```
EncoderObj *inst = (EncoderObj *)handle;
```

```
inst->workBuf = memTab[1].base;
```

1701

Figure 17

19/30

```
#define NUMBUF 3 /* max number of my memory requests */
extern IALG_Fxns *subAlg; /* sub-algorithm used by this alg */
```

```
180 { Int algNumAlloc(Void)
    {
        return (NUMBUF + subAlg->algNumAlloc());
    }
}
```

```
Int algAlloc(const IALG_Params *p, struct IALG_Fxns **pFxn,
             IALG_MemRec memTab)
```

```
{
    Int n;

    /* initialize my memory requests */
    ...
    /* initialize sub-algorithm's requests */
    n = subAlg->algAlloc(..., memTab);
    return (n + NUMBUF);
}
```

Figure 18

```
-----*/
/* TYPES AND CONSTANTS */
/*-----*/
#define IRTC_ENTER 0
#define IRTC_CLASS1 1
#define IRTC_CLASS2 2
#define IRTC_CLASS3 3
#define IRTC_CLASS4 4
#define IRTC_CLASS5 5
#define IRTC_CLASS6 6
#define IRTC_CLASS7 7
```

```
/*
 * ===== IRTC_Handle =====
 * Handle to module's trace instance object
 */
```

```
typedef struct IRTC_Obj *IRTC_Handle;
```

```
/*
 * ===== IRTC_Mask =====
 */
typedef LgUns IRTC_Mask;
```

```
/*
 * ===== IRTC_Fxns =====
 */
```

```
101 { typedef struct IRTC_Fxns {
    Void *implementationId;
    Void (*rtcBind)(LOG_Obj *log);
    IRTC_Mask (*rtcGet)(IRTC_Handle);
    Void (*rtcSet)(IRTC_Handle, IRTC_Mask mask);
} IRTC_Fxns;
```

Fig. 19

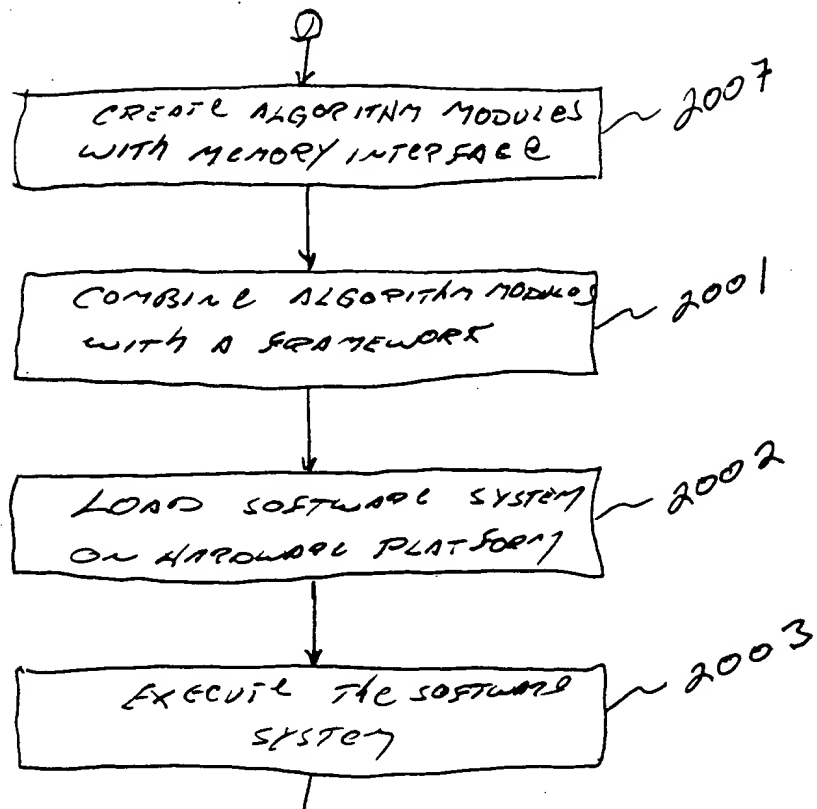


Fig 20A

21/30

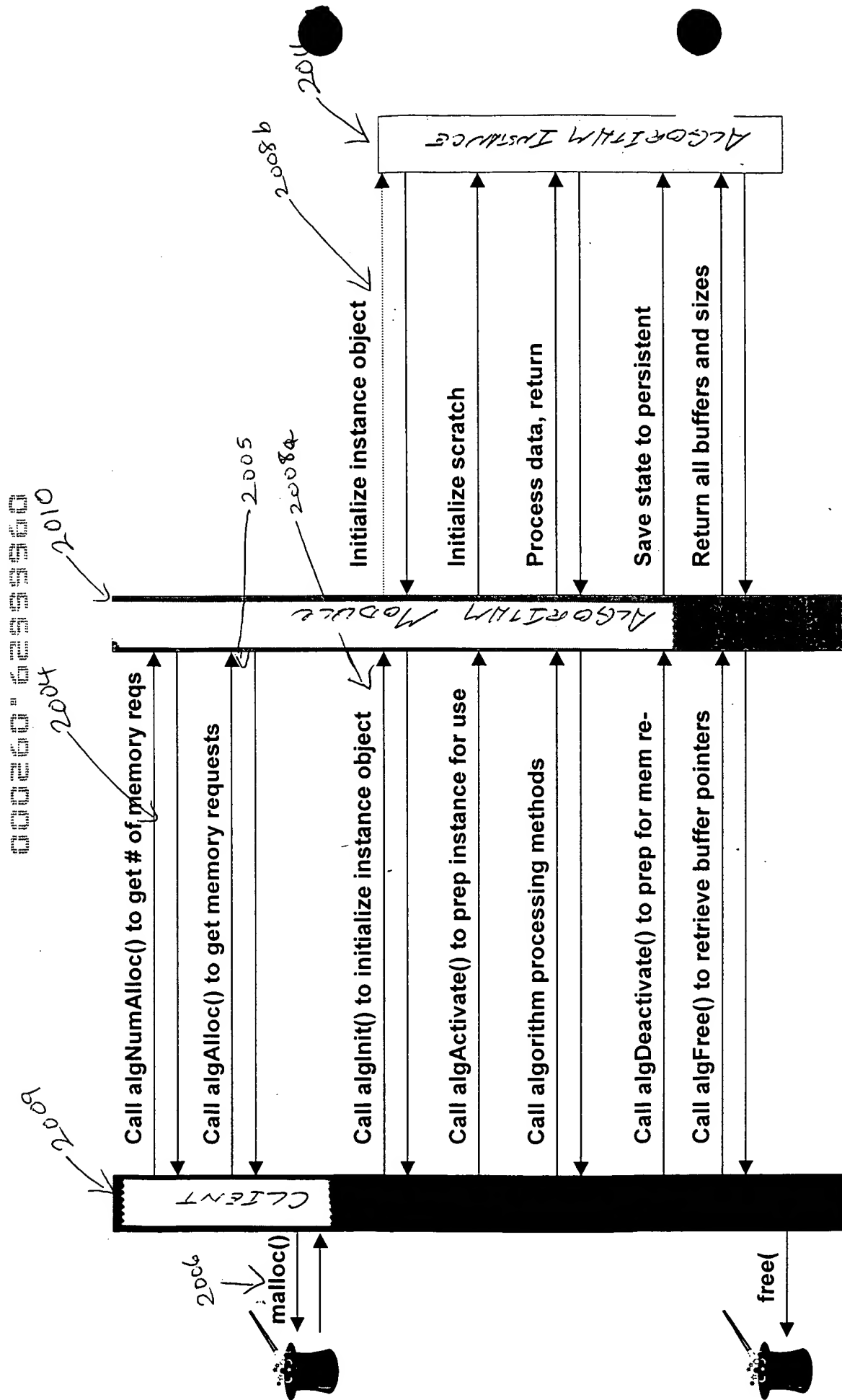


Fig. 20B

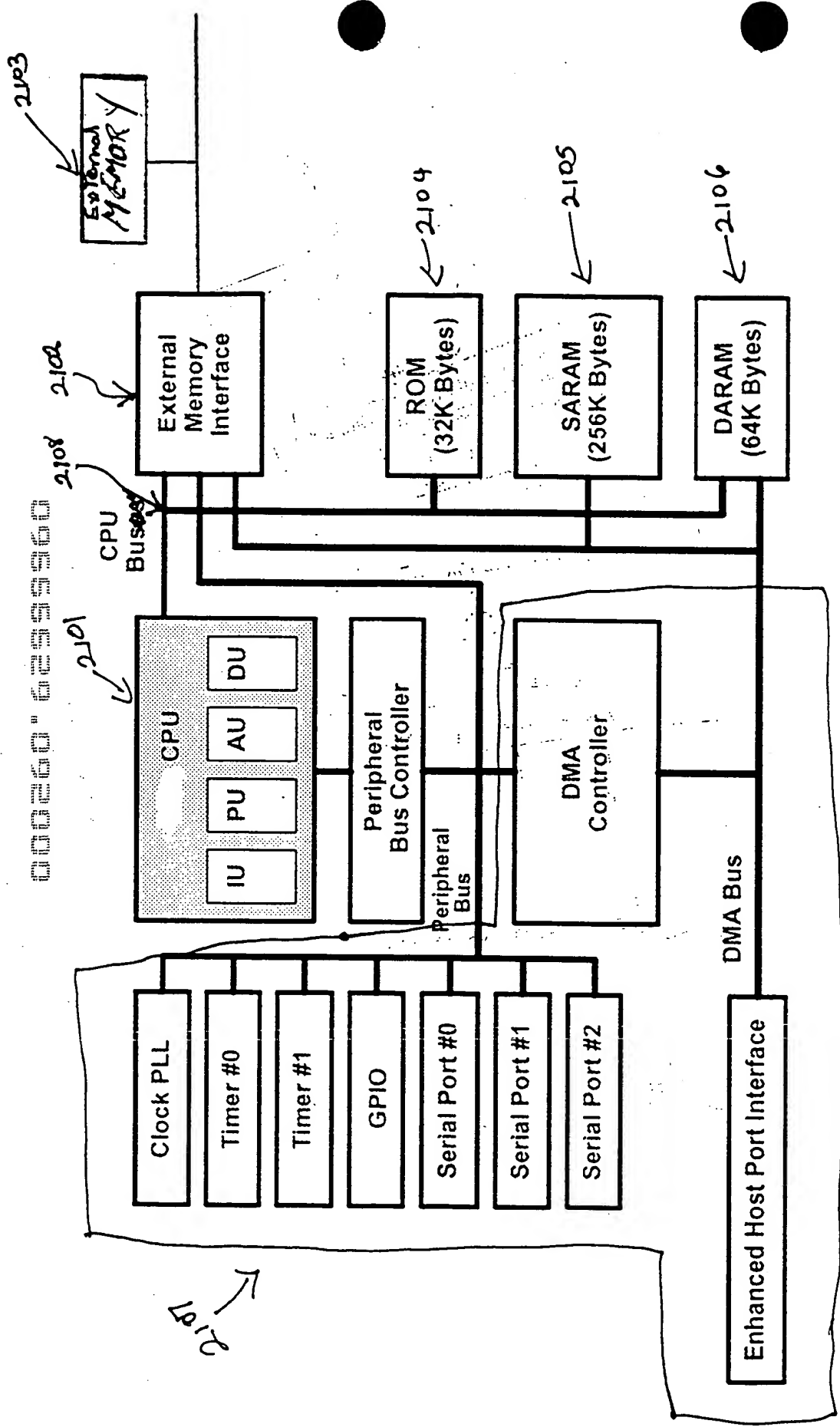
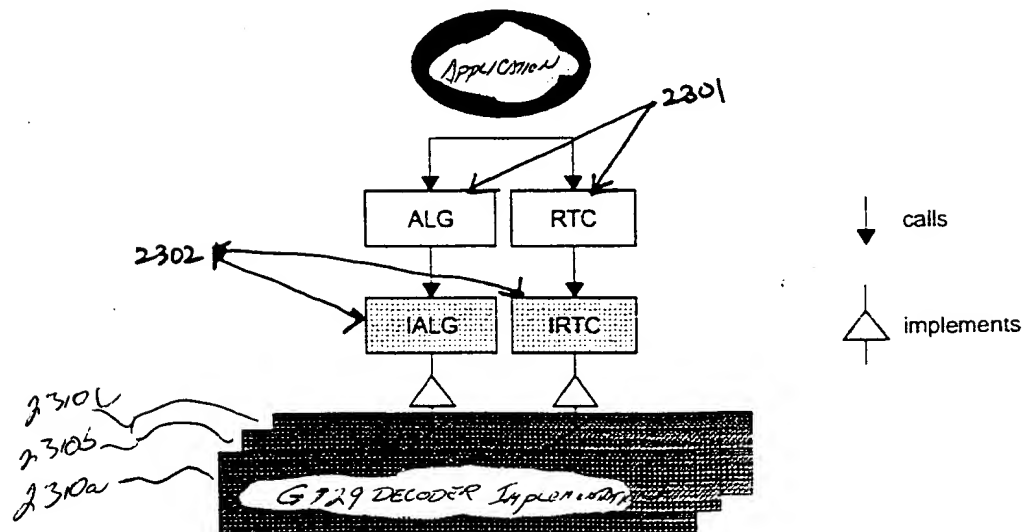
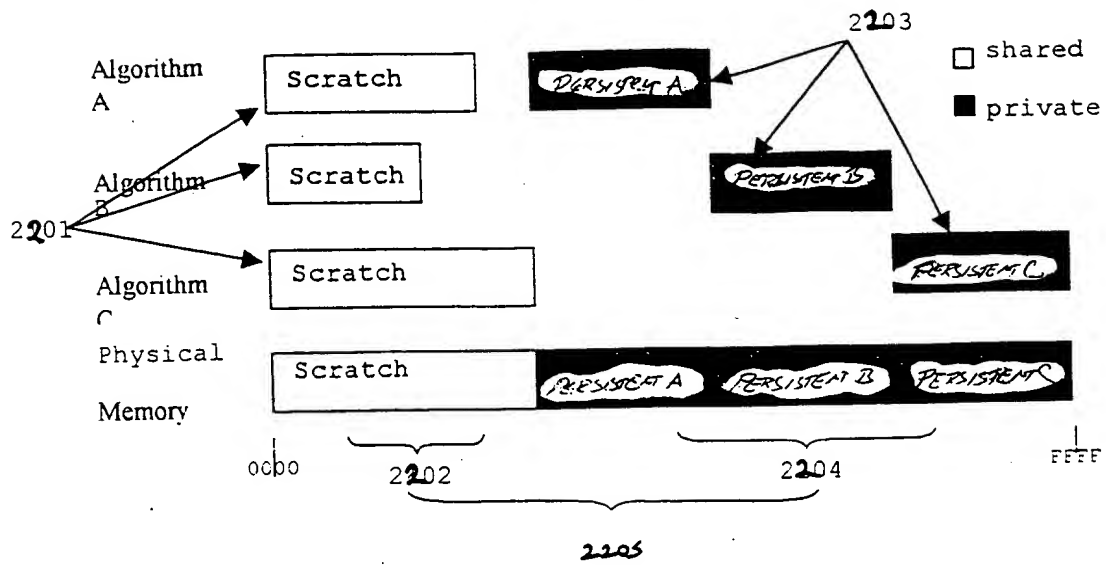


Figure 2-1



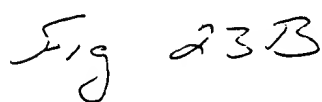


Fig 23B



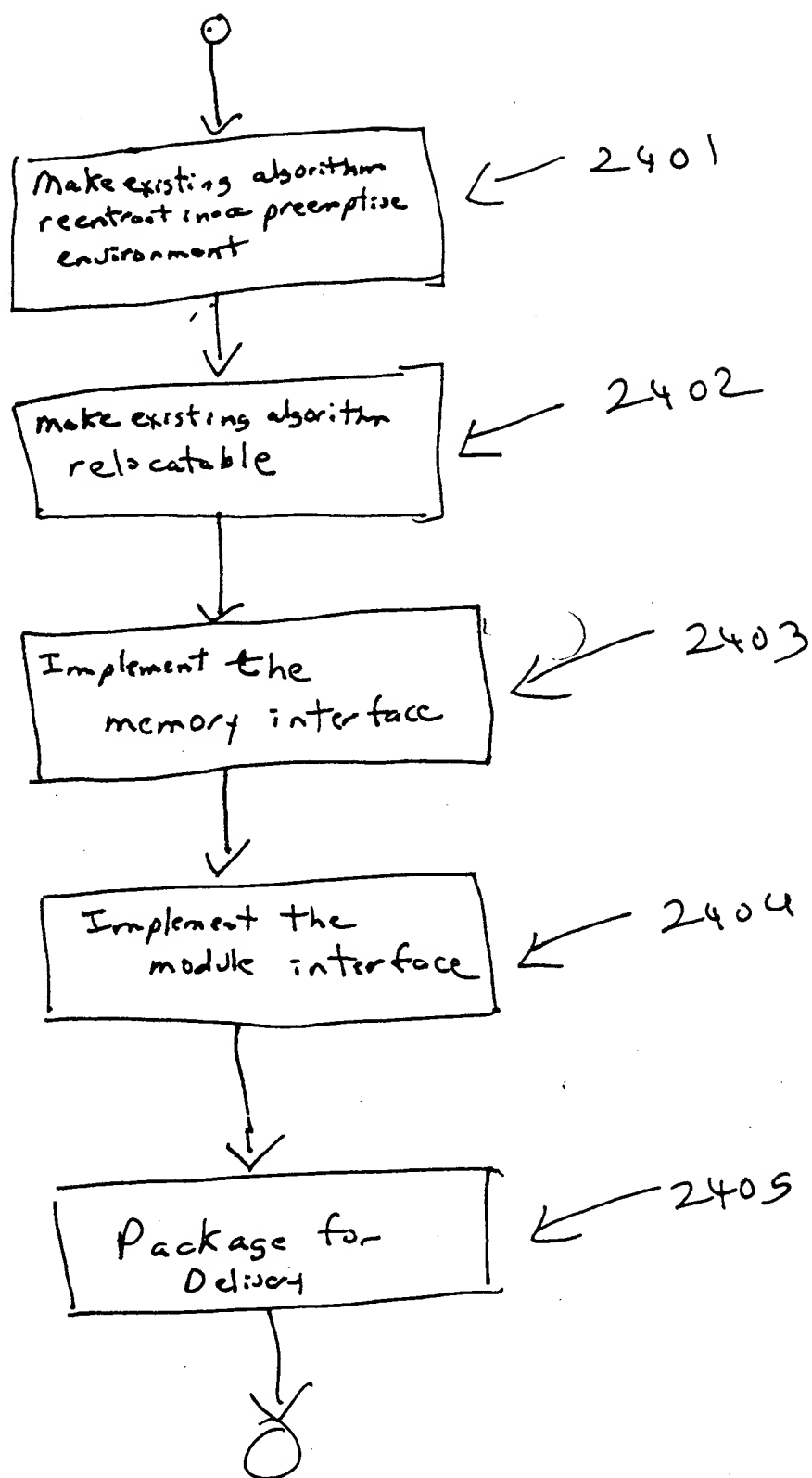
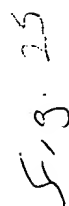


Figure 24

26/30



start state

- real-time states

27/30

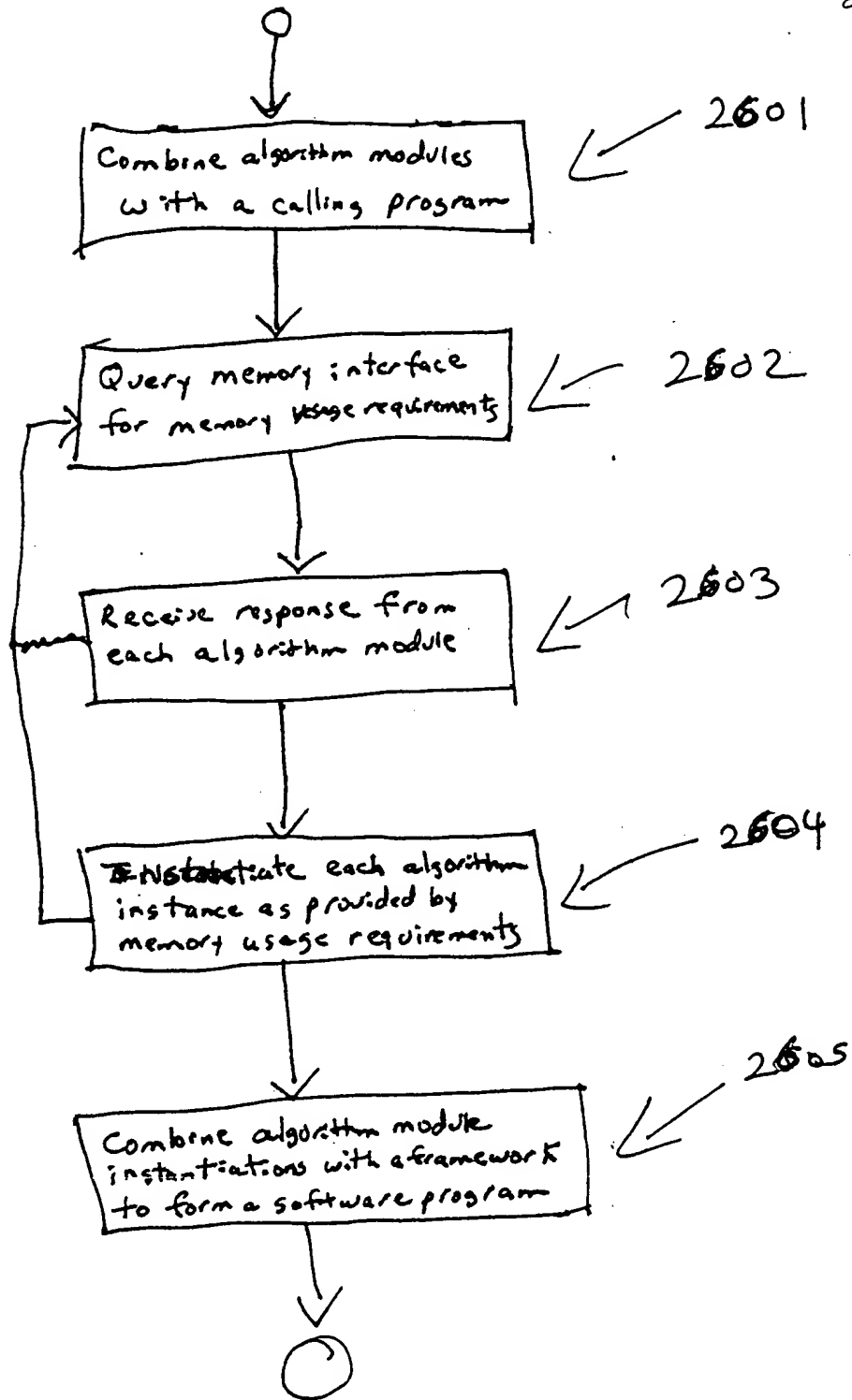
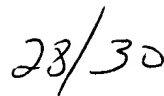


Fig. 26A

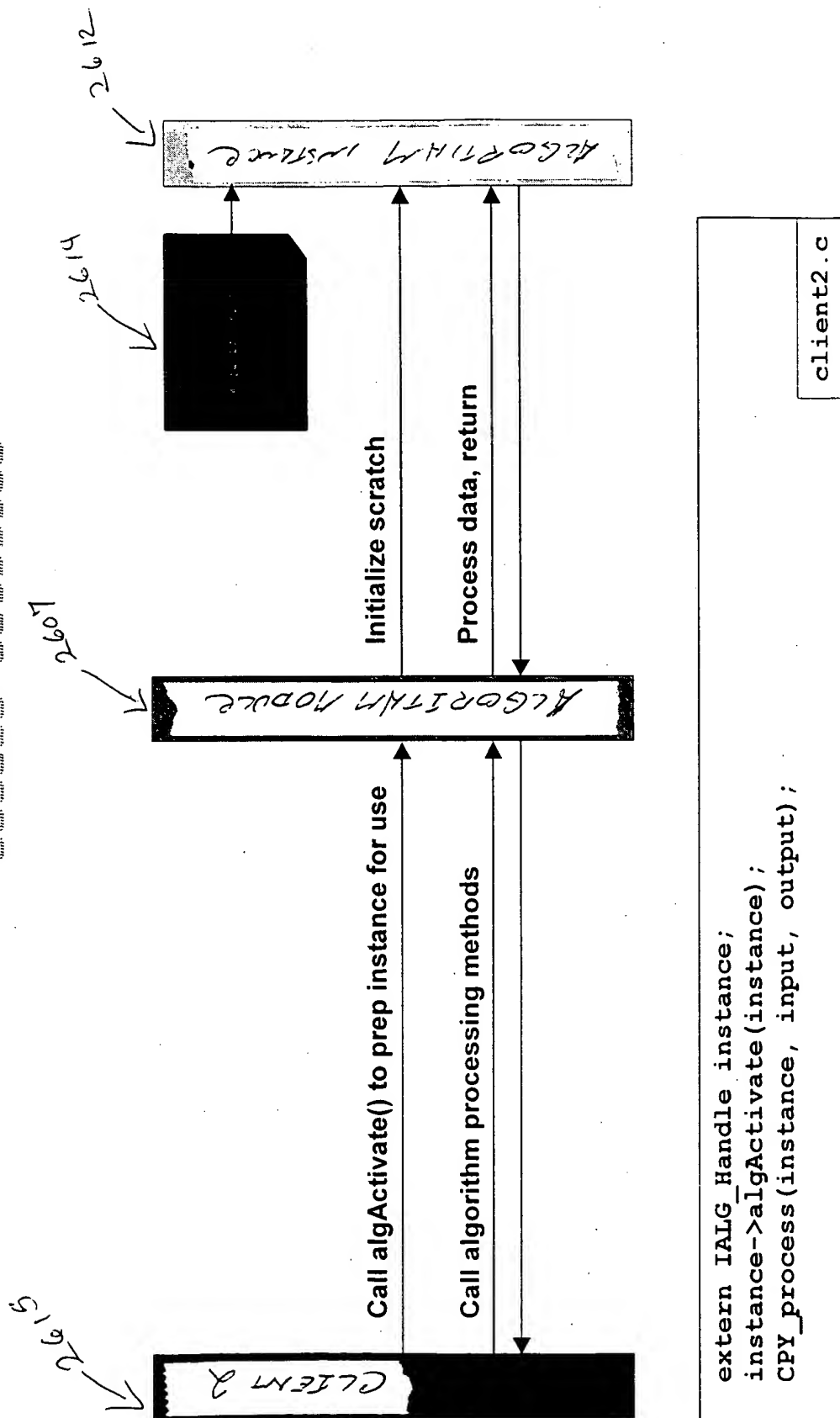
↓  
فایده



26B  
F.S.

**THE** **NEW** **YORK** **PUBLIC** **LIBRARY** **ASTOR LENOX TILDEN FOUNDATION**

29/30



Fi 260

